대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2002-0043097
Application Number

출 원 년 월 일 : 2002년 07월 23일
Date of Application JUL 23, 2002

출 원 인 : 삼성전자주식회사
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.

2003 년 07 월 07 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【제출일자】	2002.07.23
【발명의 명칭】	디지털 컨텐츠 메타데이터에 대한 복합 조건 검색 방법
【발명의 영문명칭】	MULTI-KEY SEARCH METHOD FOR DIGITAL CONTENTS METADATA
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	김동진
【대리인코드】	9-1999-000041-4
【포괄위임등록번호】	2002-007585-8
【발명자】	
【성명의 국문표기】	신효섭
【성명의 영문표기】	SHIN,Hyo Seop
【주민등록번호】	711013-1528617
【우편번호】	157-203
【주소】	서울특별시 강서구 가양3동 6단지 아파트 612동 1310호
【국적】	KR
【취지】	특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 대리인 김동진 (인)
【수수료】	
【기본출원료】	20 면 29,000 원
【가산출원료】	4 면 4,000 원
【우선권주장료】	0 건 0 원
【심사청구료】	0 항 0 원
【합계】	33,000 원
【첨부서류】	1. 요약서·명세서(도면)_1통

【요약서】

【요약】

본 발명은 TV-Anytime 포럼에서 정의하고 있는 디지털 컨텐츠 메타데이터(TVA 메타데이터)에 대한 복합 조건 검색 방법에 관한 것으로서, 특히 방송 컨텐츠정보를 표현하는 메타데이터의 멀티키 인덱싱 및 프래그먼트 XPath 인코딩 구조를 제공함으로써 메타데이터를 수신하는 셋탑박스가 프래그먼트들에 대한 복합 검색을 보다 효율적으로 수행할 수 있도록 하는 TVA 메타데이터에 대한 복합 조건 검색 방법에 관한 것이다.

본 발명은 TVA 메타데이터가 프래그먼트 단위로 전송되는 전송 스트림 환경에서, 프래그먼트의 보다 효율적인 검색 및 접근 기능을 제공하기 위한 멀티키 인덱싱 및 프래그먼트 인코딩에 대한 자료 구조 및 접근 방법을 제시함으로써 TVA 메타데이터를 수신하는 셋탑박스가 프래그먼트들에 대한 복합 검색을 멀티키 인덱싱 기법을 이용하여 보다 효율적으로 수행할 수 있게 된다.

【대표도】

도 4

【색인어】

TV-Anytime, 메타데이터, 프래그먼트, 멀티키, 인코딩

【명세서】

【발명의 명칭】

디지털 컨텐츠 메타데이터에 대한 복합 조건 검색 방법 {MULTI-KEY SEARCH METHOD FOR DIGITAL CONTENTS METADATA}

【도면의 간단한 설명】

도 1은 전자 프로그램 가이드 애플리케이션에서의 그리드 가이드 화면을 도시한다.

도 2와 도 3은 종래기술에 따른 싱글키 인덱스 검색 방법을 설명하는 도면.

도 4는 본 발명에 따른 멀티키 인덱스 검색 방법을 설명하는 도면.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<4> 본 발명은 TV-Anytime 포럼에서 정의하고 있는 디지털 컨텐츠 메타데이터(metadata)(이하, TVA 메타데이터라 함)에 대한 복합 조건 검색 방법에 관한 것으로서, 특히 방송 컨텐츠(contents) 정보를 표현하는 메타데이터의 멀티키 인덱싱 및 프래그먼트 XPath 인코딩 구조를 제공함으로써 메타데이터를 수신하는 셋탑박스가 프래그먼트들에 대한 복합 검색을 보다 효율적으로 수행할 수 있도록 하는 TVA 메타데이터에 대한 복합 조건 검색 방법에 관한 것이다.

<5> TV-Anytime 포럼은 개인용 대용량 저장매체를 갖는 사용자 환경에서 오디오 비주얼 관련 서비스 제공을 위한 표준 개발을 목적으로 하는 민간 표준 기구이며, 모든 사용자

가 개인용 저장장치를 기반으로 자기가 원하는 방법으로 원하는 시간에 다양한 형태(기존의 방송 서비스 및 온라인 대화형 서비스 등)의 프로그램을 시청할 수 있게 하는 것을 그 구체적인 서비스 목표로 하고 있다.

<6> 메타데이터는 프로그램 타이틀과 개요와 같은 컨텐츠에 대한 설명적인 데이터를 의미하며, "데이터에 관한 데이터(data about data)"로 정의된다. TV-Anytime 포럼의 메타데이터 규격에서는 XML Schema 언어를 사용하여 그 구조를 정의하는 한편, 각 메타데이터 엘리먼트 및 속성에 대한 의미(semantics)를 함께 규정하고 있다. 방송 컨텐츠에 관련된 주요 메타데이터에는 예를 들어 프로그램 정보, 그룹 정보, 프로그램 리뷰 정보, 크레디트 정보, 세그먼트 정보, 프로그램 위치 정보, 서비스 정보 등이 있다. 이러한 메타데이터들은 방송국에서 사용자 셋탑박스로의 전송 스트림상에서 프래그먼트(fragment)라는 독립적인 단위로 전송되어지며, 따라서 프래그먼트들에 대한 효율적인 검색과 접근을 제공하기 위해서는 메타데이터 인덱싱 구조가 필요하다.

<7> 종래기술문헌(TV-Anytime Specification TV145, J.P. Evain, "1st Draft of Metadata Specification SP003v1.3," TV-Anytime 포럼 17차 회의, 몬트리올, 캐나다, 202년 6월)은 방송국에서 사용자 셋탑박스로의 전송 스트림상에서 TVA 메타데이터의 프래그먼트를 검색하고 접근하기 위한 싱글키 인덱싱 구조를 제안하고 있다. 이 문헌에서, 메타데이터 프래그먼트는 아래에 설명된 인덱싱 시스템을 사용하여 배치되고 식별되어진다. 또한, 모든 데이터가 전송되는 최상위 저장소를 형성하는 컨테이너(container) 개념이 사용되는데, 이 컨테이너는 key_index_list,

key_index, sub_key_index, string_repository, fragment_data_repository를 운반하는 인덱스 컨테이너와 elements_table, string_repository, fragment_data_repository를 운반하는 데이터 컨테이너로 구분된다.

<8> 키 인덱스 리스트

<9> 키 인덱스 리스트(key_index_list)는 전송되는 모든 인덱스의 리스트를 제공한다. 각 인덱스는 프래그먼트에 대한 xpath(fragment_xpath_ptr)와 프래그먼트를 검색하기 위한 키 필드 xpath(key_xpath_ptr)로 구분되어 지정된다. 여기서, xpath는 문서내에서 하나 이상의 노드에 대한 경로를 기술할 수 있는 문장을 의미한다.

<10> 【표 1】

문장	비트수	식별자
key_index_list() {		
for (j=0; j<key_index_count; j++) {		
fragment_xpath_ptr	16	ushort
key_xpath_ptr	16	ushort
key_encoding_indicator	2	
unused	6	
if (key_encoding_indicator == '10') {		
key_encoding	8	ubyte
} else {		
unused	16	
}		
index_container	16	ushort
key_index_identifier	8	ubyte
}		
}		

<11> fragment_xpath_ptr: 프래그먼트 xpath 스트링의 시작에 대한 레퍼런스이다. 이 레퍼런스는 현재 컨테이너내의 스트링 저장소의 시작으로부터 바이트당 오프셋의 형태로 있다.

<12> key_xpath_ptr: 현재 컨테이너에 속하는 스트링 테이블내의 키 xpath 스트링의 시작에 대한 레퍼런스. 이 스트링 값은 인덱스 키로서 사용되는 노드의 fragment_xpath_ptr에 관련한 경로이다.

<13> key_encoding_indicator: 키 인덱스 섹션의 high_key_value와 서브 키 인덱스 섹션의 key_value 부재에 대해 사용된 인코딩 존재를 나타낸다.

<14> key_encoding: 인코딩 방법 설명 스트링의 시작에 대한 레퍼런스.

<15> index_container: 목표 key_index 구조의 container_id.

<16> **키 인덱스**

<17> key_index 구조는 위에서 설명한 key_index_list에서 명세되어 있는 각 인덱스의 루트 블록을 나타낸다. key_index는 이후에 설명되는, 실제 키 값을 지정하는 sub_key_index 구조에 대한 포인터의 리스트를 가지고 있으며, 가장 큰 키 값 (high_key_value)을 기준으로 sub_key_index를 지정한다.

<18>

【표 2】

문장	비트수	식별자
key_index() {		
key_index_identifier	8	ubyte
for (j=0; j<sub_index_count; j++) {		
high_key_value	16	ushort
sub_index_container	16	ushort
sub_index_identifier	8	ubyte
}		
}		

<19> key_index_identifier: 하나 이상의 key_index을 유지하는 컨테이너내의 key_index structure를 식별한다.

<20> high_key_value: 소정 서브 인덱스에 의해 참조될 수 있는 가장 큰 키 값.

<21> sub_index_container: 서브 인덱스 섹션을 운반하는 컨테이너의 ID.

<22> sub_index_identifier: 목표 sub_key_index 구조의 sub_index_identifier 필드의 값.

<23> 서브 키 인덱스

<24> 서브 키 인덱스(sub_key_index)는 해당 키 값(key_value)을 가지는 프래그먼트의 주소(target_container, target_handle)를 지정한다.

<25>

【표 3】

문장	비트수	식별자
sub_key_index() {		
sub_index_identifier	8	ubyte
for (j=0; j<reference_count; j++) {		
key_value	16	ushort
target_container	16	ushort
target_handle	16	ushort
}		
}		

<26> sub_key_identifier: 하나 이상의 sub_key_index를 유지하는 컨테이너내의 sub_key_index 구조를 식별한다.

<27> 이러한 종래기술문헌에서는 TV-Anytime 규격의 메타데이터 프래그먼트에 대한 특정 필드를 키로 하는 인덱스 구조를 제공하여, 캐루젤(carousel) 안에서 접근하고자 하는 프래그먼트를 검색하는 기능을 제공한다.

<28> 그러나, 싱글키 인덱싱 검색만을 지원하기 때문에 하나 이상의 질문 조건 즉, 다중 필드 값에 대한 메타데이터 프래그먼트에 대한 검색과 접근을 효율적으로 처리하지 못한다는 단점을 가진다. 예를 들면, 셋탑박스의 전자 프로그램 가이드(Electronic Programs Guide : EPG) 애플리케이션에서, 도 1과 같은 그리드 가이드(grid guide) 화면을 구성한다고 하자. 그리드 가이드는 세로축은 채널, 가로축은 시간을 기준으로 격자형으로 구성된다. 그리드 가이드에 프로그램 리스트를 표시하기 위해서는 2개 필드 즉, 채널 번호

및 방송 시간에 대한 검색 동작이 요구된다. TV-Anytime 메타데이터 용어에서, 이러한 검색 동작은 다음과 같이 표현된다.

<29> - 목표 프래그먼트 검색 (Program Location):

<30> /TVAMain/ProgramDescription/ProgramLocationTable/BroadcastEvent

<31> - 키 리스트:

<32> ServiceId, EventDescription/PublishedTime

<33> 싱글키 인덱싱 구조에서, 검색 키에 의해 지정된 검색 조건에 부합하는 프래그먼트를 얻기 위해서는 2가지 방법이 가능한데, 이러한 방법에 대해 도 2와 도 3를 참조하여 설명하면 다음과 같다.

<34> 한가지 방법은 도 2에 설명된 바와 같이 중간 목표 프래그먼트가 ServiceId와 EventDescription/PublishedTime에 대한 검색 키 인덱스중 하나를 사용함으로써 서로 독립적으로 검색되고, 다음에 중간 프래그먼트 세트를 교차시킴으로써 최종결과가 얻어지는 것이다.

<35> 단계 S10 및 S12에서, 채널 필드 즉, 채널번호를 키로 하는 인덱스를 접근하여 해당 채널 범위의 프래그먼트를 검색하여 가져온다. 즉, key_index_list 에서,

<36> fragment_xpath_ptr=

<37> /TVAMain/ProgramDescription/ProgramLocationTable/BroadcastEvent

<38> key_xpath_ptr = ServiceId가 되는 인덱스를 지정하여 검색한다.

<39> 단계 S20 및 S22에서, 시간 필드 즉, 방송시간을 키로 하는 인덱스를 접근하여 해당 시간 범위의 프래그먼트를 검색하여 가져온다. 즉, key_index_list 에서,

<40> fragment_xpath_ptr=

<41> /TVAMain/ProgramDescription/ProgramLocationTable/BroadcastEvent

<42> key_xpath_ptr = EventDescription/PublishedTime가 되는 인덱스를 지정하여 검색한다.

<43> 단계 S30에서, 상기 단계들에 의해서 독립적으로 검색된 중간 결과 프래그먼트들중 공통으로 들어있는 프래그먼트를 골라낸다.

<44> 다른 한가지 방법은 도 3에 설명된 바와 같이 2개의 싱글키 인덱스중 하나(예를 들어, ServiceId에 대한)만을 사용하여 목표 프래그먼트를 검색하고(단계 S40 및 단계 S42), 그후 중간 프래그먼트가 다른 검색 키 조건(예를 들어, PublishedTime)과 대조하여 필터링을 수행한다(단계 S44).

<45> 그러나, 이러한 방법들은 싱글키 인덱스로 참조됨으로써 초래되는 중간 결과의 크기가 통상적으로 커지기 때문에 전체 프로세스가 효율적이지 못하다. 즉, ServiceId 또는 PublishedTime에 대한 필터링은 상당한 수의 프래그먼트를 생성한다. 첫번째 방법에서는 프로그램 시간대에 관계없이 해당 채널 범위의 모든 프로그램을 검색 결과로 가져오기 때문에 검색 결과가 매우 클 수 있으며, 또한 모든 채널에 대하여 해당 시간대의 프로그램을 검색하기 때문에 검색 결과가 매우 클 수 있다. 더욱이, 크기가 큰 두 가지의 중간 검색 결과를 병합하는 과정에서도 계산 복잡도가 매우 커지게 된다. 두번째 방법에서는 하나의 큰 중간 결과가 다른 검색 키 조건과 대비하여 추가로 제거되어야 한다. 결과적으로, 셋탑 박스 안에서 많은 오버헤드가 발생할 수 있는 문제의 여지가 있는 것이다.

<46> 한편, 다른 문제점은 key_index_list 구조에서 각 프래그먼트를 지정함에 있어서 xpath를 그대로 사용한다는 점이다. xpath는 기본적으로 정규 경로식(regular expression)으로 표현되므로, 동일한 프래그먼트를 지정함에 있어서도 여러 가지 표현이 가능하기 때문에 셋탑 박스 안에서 xpath에 대한 프래그먼트를 지정하는 과정에서 또다른 오버헤드를 유발할 수 있다. 또한, TVA 메타데이터에서는 주요 프래그먼트 타입들이 미리 정의되어 있기 때문에, 미리 지정되는 xpath에 의한 저장 공간의 오버헤드가 발생할 수도 있다.

<47> 따라서, ServiceId와 PublishedTime, ProgramID(CRID)와 SegmentID, GroupID와 PublishTime에 대한 멀티키 인덱스가 존재한다면 2개의 검색 조건에 기초한 목표 프래그먼트 검색 처리가 더욱 용이할 것이다.

【발명이 이루고자 하는 기술적 과제】

<48> 상술한 문제점을 해결하기 위해 제안된 본 발명은, 디지털 컨텐츠 메타데이터의 프래그먼트 검색과 접근에 유용한 멀티키 인덱스로서 TV-Anytime 시스템 트랜스포트 계층에서 사용될 멀티키 인덱싱의 인코딩 개념을 제공하는데 그 목적이 있다.

<49> 본 발명의 다른 목적은 더욱 효율적인 방식으로 인덱싱되는 목표 프래그먼트 식별에 대한 인코딩 개념을 제공하는 것이다.

【발명의 구성 및 작용】

<50> 상술한 목적을 달성하기 위한 본 발명에 따른 방법은 프래그먼트의 정보를 나타내는 프래그먼트 기술자(fragment_descriptor)와 멀티키 인덱스를 위한 멀티키 정보를 나타내는 키 기술자(key_descriptor)를 가지는 키 인덱스 리스트(key_index_list) 구조를

정의하는 단계; 및 입력된 질의를 분석하여 멀티키 검색일 경우 상기 정의된 키 인덱스 리스트(key_index_list) 구조를 이용하여 멀티키 인덱스를 식별하며, 키 인덱스(key_index) 및 서브 키 인덱스(sub_key_index) 구조를 이용하여 멀티키 인덱스 검색을 수행하는 단계를 포함하는 것을 특징으로 한다.

<51> 또한, 상기 키 인덱스 리스트 구조를 정의하는 단계는, 빈번히 참조되는 프래그먼트 타입에 대해 지정된 인코딩 값을 부여하는 단계를 포함하는 것을 특징으로 한다.

<52> 따라서, TV-Anytime 메타데이터를 수신하는 셋탑박스가 프래그먼트들에 대한 복합 검색을 멀티키 인덱싱 기법을 이용하여 보다 효율적으로 수행할 수 있게 된다.

<53> 이하, 본 발명의 바람직한 일실시예를 상세하게 설명한다.

<54> 본 발명에서는 TVA 메타데이터 프래그먼트에 대한 멀티키 인덱싱 구조를 프래그먼트 xpath 인코딩 구조와 함께 제시한다. 본 발명에 따른 메타데이터 프래그먼트 인덱싱 구조는 key_index_list(), key_index(), sub_key_index()와 같은 재편성된 구조 및 fragment_descriptor(), key_descriptor(), high_key_value_descriptor(), key_value_descriptor()와 같은 새롭게 도입된 구조를 포함하고 있다.

<55> **1. 키 인덱스 리스트(key index list)**

<56> 키 인덱스 리스트는 전체 XML 문서에 대해 이용가능한 모든 인덱스를 나열한다. 각각의 인덱스 구조는 표 4에 제시된 바와 같이 프래그먼트 xpath 인코딩 및 멀티키 인덱싱이 가능하도록 fragment_descriptor(), key_descriptor()를 새롭게 가진다.

<57>

【표 4】

신택스	비트수	식별자
key_index_list() {		
key_index_count	8	ushort
for (j=0; j<key_index_count; j++) {		
fragment_descriptor()		
key_descriptor()		
index_container	16	ushort
key_index_identifier	8	ubyte
}		
}		

<58> key_index_count: 전체 XML 문서에 대한 인덱스 수를 상술함.

<59> fragment_descriptor(): XML 문서내의 프래그먼트의 타입 또는 프래그먼트의 XPath 위치와 같은 인덱싱될 프래그먼트 그룹의 정보를 기술한다.

<60> key_descriptor(): 인덱싱될 목표 프래그먼트 그룹의 XPath 위치내에서 관련 XPath 위치와 같은 인덱스를 위한 멀티키 및 상기 멀티키를 구성하는 각 엘리먼트/속성에 대한 인코딩 표시자의 정보를 기술한다.

<61> index_contaniner: 지정된 인덱스가 존재하는 컨테이너를 식별한다.

<62> key_index_identifier: index_contaniner에 의해 상술된 컨테이너내에서 키 인덱스 섹션을 식별한다. index_contaniner와 key_index_identifier의 조합에 의해 키 인덱스 섹션이 유일하게 식별될 수 있다.

<63> 2. 프래그먼트 디스크립터(fragment_descriptor)

<64> fragment_descriptor()는 인덱싱되는 목표 프래그먼트의 타입을 식별한다. 표 5는 fragment_descriptor()의 구조를 설명하고 있다.

<65> 【표 5】

신택스	비트수	식별자
fragment_descriptor() {		
fragment_type	8	ubyte
if (fragment_type == 0x0F) {		
fragment_xpath_ptr	16	ushort
}		
}		

<66> fragment_type: 인덱싱될 프래그먼트 그룹의 타입을 나타낸다. 미리 정의된 값은 빈번히 참조되는 프래그먼트 그룹에 대해 인코딩된다. 프래그먼트 그룹이 미리 정의된 타입이 아니라면(사용자가 지정한 경우), 0x0F로서 인코딩되고 이전과 같이 fragment_xpath_ptr가 주어진다.

<67> 아래의 표 6은 TVA 문서에서 빈번히 참조되는 프래그먼트 타입(fragment_type)에 대한 각각의 인코딩 값을 나타낸다. 이러한 프래그먼트 타입(0x01~0x07)은 EPG 애플리케이션에서 빈번히 검색되고 인덱싱될 것으로 예측되는 것이다.

<68>

【표 6】

값	설명
0x01	Program Information
0x02	Group Information
0x03	Credits Information
0x04	Program Review
0x05	Segment Information
0x06	Service Information
0x07	Broadcast Event
0x0F	User Private
0x08-0x0E 0x10-0xFF	Reserved

<69> 이러한 프래그먼트 타입에 대한 인코딩 개념은 빈번히 참조되는 프래그먼트 타입의 표현 크기를 감소시킬 뿐만 아니라 이들이 나타내는 프래그먼트 그룹을 식별하기 위한 fragment_xpath_ptr의 분석에서 발생되는 셋탑 박스의 오버헤드를 경감시킨다.

<70> 3. 키 디스크립터(key descriptor)

<71> 멀티키는 복합의 키 속성이다. key_descriptor는 인덱싱되는 프래그먼트 타입내의 관련 xpath 위치 및 인코딩 지시자와 같은 각각의 키 속성에 대한 특성을 기술한다. 아래의 표 7은 key_descriptor를 나타낸다.

<72>

【표 7】

신택스	비트 수	식별자
key_descriptor() {		
key_attribute_count	8	ushort
for (j=0; j<key_attribute_count; j++) {		
key_xpath_ptr	16	ushort
key_encoding_indicator	2	
unused	6	
if (key_encoding_indicator == '10') {		
key_encoding	8	ubyte
} else {		
unused	16	
}		
}		
}		

<73> key_attribute_count: 멀티키를 구성하는 키 속성의 수를 상술한다. 단일키 인덱스에 대해서도, 멀티키 인덱싱 구조는 단일키 인덱싱 구조에 8비트만을 부가함에 유의하라.

<74> 4. 키 인덱스(Key Index)

<75> key_index() 구조내에서, high_key_value_descriptor()가 멀티키 인덱싱 구조에 대해 새롭게 도입된다. high_key_value_descriptor()는 서브 인덱스 섹션에 대한 가장 큰 멀티키값 포인팅을 나타내는 키 속성값의 조합이다.

<76>

【표 8】

신택스	비트수	식별자
key_index() {		
key_index_identifier	8	ubyte
sub_index_count	8	ushort
for (j=0; j<sub_index_count; j++) {		
high_key_value_descriptor()	16 * key_attribute_count	ushort
sub_index_container	16	ushort
sub_index_identifier	8	ubyte
}		
}		

<77> 표 9는 high_key_value_descriptor()를 나타낸다.

<78> 【표 9】

신택스	비트수	식별자
high_key_value_descriptor() {		
for (j=0; j<key_attribute_count; j++) {		
key_attribute_value	16	ushort
}		
}		

<79> key_attribute_count: 멀티키를 구성하는 키 속성의 수를 상술한다. 이것은 키 인덱스 리스트 섹션에서 정의된다.

<80> key_attribute_value: 각각의 키 속성에 대한 값을 나타낸다. 상기 값 인코딩 형태는 종래의 단일키 인덱싱 구조의 key_value와 같다.

<81> 멀티키 값 사이의 비교와 관련하여, 멀티키 값은 사전 편찬 방식에 의해 순서가 결정된다. 예를 들면, 멀티키 (k_1, k_2, ..., k_n)에 대해, k_1이 가장 큰 우선순위를 가지고 k_n은 가장 작은 우선순위를 가진다. 두 멀티키값 (a_1, a_2, ..., a_n) 와 (b_1, b_2, ..., b_n)에 대하여, 모든 j ($0 \leq j \leq i-1$)에 대해 $a_j=b_j$ 및 $a_i>b_i$가 되도록 정수 i ($0 \leq i \leq n-1$)가 존재하는 경우에 한해서 (a_1, a_2, ..., a_n) 가 (b_1, b_2, ..., b_n)보다 더 크다. 유사한 방식으로, 모든 j ($0 \leq j \leq i-1$)에 대해 $a_j=b_j$ 및 $a_i<b_i$가 되도록 정수 i ($0 \leq i \leq n-1$)가 존재하는 경우에 한해서 (a_1, a_2, ..., a_n)가 (b_1, b_2, ..., b_n)보다 더 작다. 모든 i ($1 \leq i \leq n$)에 대해 $a_i=b_i$인 경우에 한해서 (a_1, a_2, ..., a_n)는 (b_1, b_2, ..., b_n)과 같다.

<82> 5. 서브 키 인덱스(Sub Key Index)

<83> sub_key_index 구조에서, key_value_descriptor()가 멀티 인덱싱 개념을 위해 새롭게 도입된다. key_value_descriptor()는 목표 프래그먼트의 멀티키 값 포인팅을 나타내는 키 속성값의 조합이다.

<84>

【표 10】

신택스	비트수	식별자
sub_key_index() {		
sub_index_identifier	8	ubyte
reference_count	8	ushort
for (j=0; j<reference_count; j++) {		
key_value_descriptor()	16 * key_attribute_count	ushort
target_container	16	ushort
target_handle	16	ushort
}		
}		

<85> 표 11은 key_value_descriptor()를 나타낸다.

<86> 【표 11】

신택스	비트수	식별자
key_value_descriptor() {		
for (j=0; j<key_attribute_count; j++) {		
key_attribute_value	16	ushort
}		
}		

<87> key_attribute_count: 멀티키를 구성하는 속성의 수를 상술하며, 키 인덱스 리스트 섹션에서 정의된다.

<88> key_attribute_value: 각각의 키 속성을 표현한다. 그 형태는 종래의 단일키 인덱싱 구조에서의 key_value와 동일하다.

<89> key_value_descriptor() 사이의 비교는 키 인덱스 섹션 구조에서의 high_key_value_descriptor() 사이의 비교와 동일하므로 이에 대한 설명은 생략한다.

<90> **6. 멀티키 인덱싱을 통한 프래그먼트 검색**

<91> 도 4을 참조하여, 종래기술의 문제점을 설명하면서 언급한 그리드 가이드에서 <채널, 시간>에 대한 프래그먼트 검색을 예로 들어보면, <채널,시간>에 대한 멀티키 인덱스에 의해 프래그먼트가 검색되는 과정은 다음과 같다.

<92> (1) 멀티키 인덱스 선택(단계 S100)

<93> 현재 전송되어지는 메타데이터 프래그먼트들에 대한 key_index_list() 섹션을 접근한다. key_index_list()내에 묘사된 각각의 인덱스 구조들 중에서, fragment_descriptor()안의 fragment_type이 Broadcast Event를 나타내는 0x07 이고, key_descriptor()의 key_attribute_count가 2이며, 각각의 key attribute의 key_xpath_ptr가 ServiceId, EventDescription/PublishedTime인 멀티키 인덱스를 선택한다. 즉, 해당 key_index 섹션을 지정한다.

<94> (2) 멀티키를 이용한 복합 조건 검색(단계 S110)

<95> key_index()는 sub_key_index()를 지정하고, 이어서 sub_key_index()는 목표 프래그먼트를 지정한다. 이러한 두단계 과정은 본 발명에서 앞서 설명한 바와 같은 멀티키 비교방식에 의거한다. 즉, key_index()에서 sub_key_index()를 지정한 경우에는 high_key_value_descriptor() 값이 검색 조건값(채널번호와 방송시간) 보다 작거나 같은 것들중 가장 큰 것을 선택한다. 또한, sub_key_index()에서 목표 프래그먼트를 지정할

때에는 key_value_descriptor() 값이 검색조건과 맞는 것을 선택한다. 최종적으로, 인덱스 검색의 결과로는 검색조건에 맞는 프래그먼트에 대한 식별자가 반환된다.

<96> 이와 같이, 본 발명에 따른 멀티키 인덱스 검색방법에서는 여러 번의 싱글키 인덱스 접근과 검색 결과의 병합 과정은 요구되지 않는다.

【발명의 효과】

<97> 본 발명은 TV-Anytime 메타데이터가 프래그먼트 단위로 전송되는 전송 스트림 환경에서, 프래그먼트의 보다 효율적인 검색 및 접근 기능을 제공하기 위한 멀티키 인덱싱 및 프래그먼트 인코딩에 대한 자료 구조 및 접근 방법을 제시하였다.

<98> 본 발명을 통하여, TV-Anytime 메타데이터를 수신하는 셋탑박스가 프래그먼트들에 대한 복합 검색을 멀티키 인덱싱 기법을 이용하여 보다 효율적으로 수행할 수 있게 된다

【특허청구범위】

【청구항 1】

프래그먼트의 정보를 나타내는 프래그먼트 기술자(fragment_descriptor)와 멀티키 인덱스를 위한 멀티키 정보를 나타내는 키 기술자(key_descriptor)를 가지는 키 인덱스 리스트(key_index_list) 구조를 정의하는 단계; 및

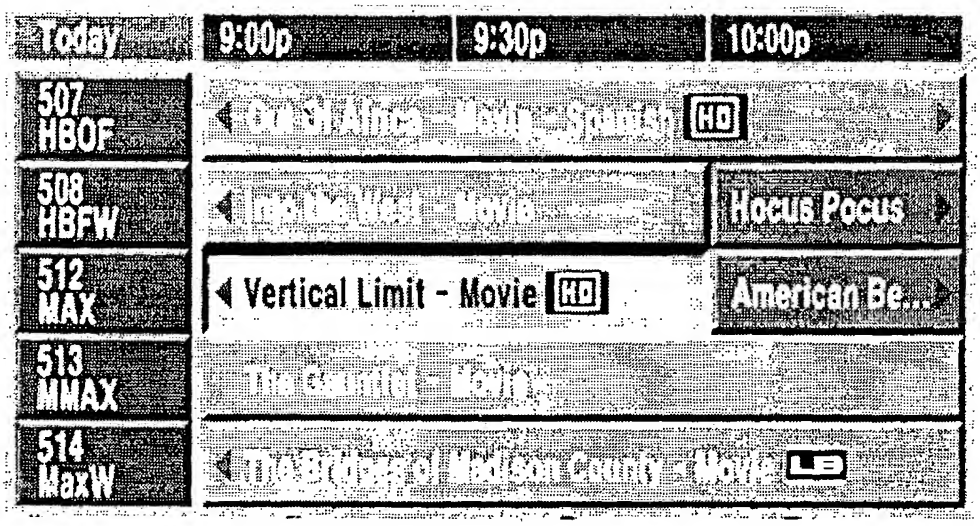
입력된 질의를 분석하여 멀티키 검색일 경우 상기 정의된 키 인덱스 리스트(key_index_list) 구조를 이용하여 멀티키 인덱스를 식별하며, 키 인덱스(key_index) 및 서브 키 인덱스(sub_key_index) 구조를 이용하여 멀티키 인덱스 검색을 수행하는 단계를 포함하는 것을 특징으로 하는 디지털 컨텐츠 메타데이터 프래그먼트에 대한 복합 조건 검색 방법.

【청구항 2】

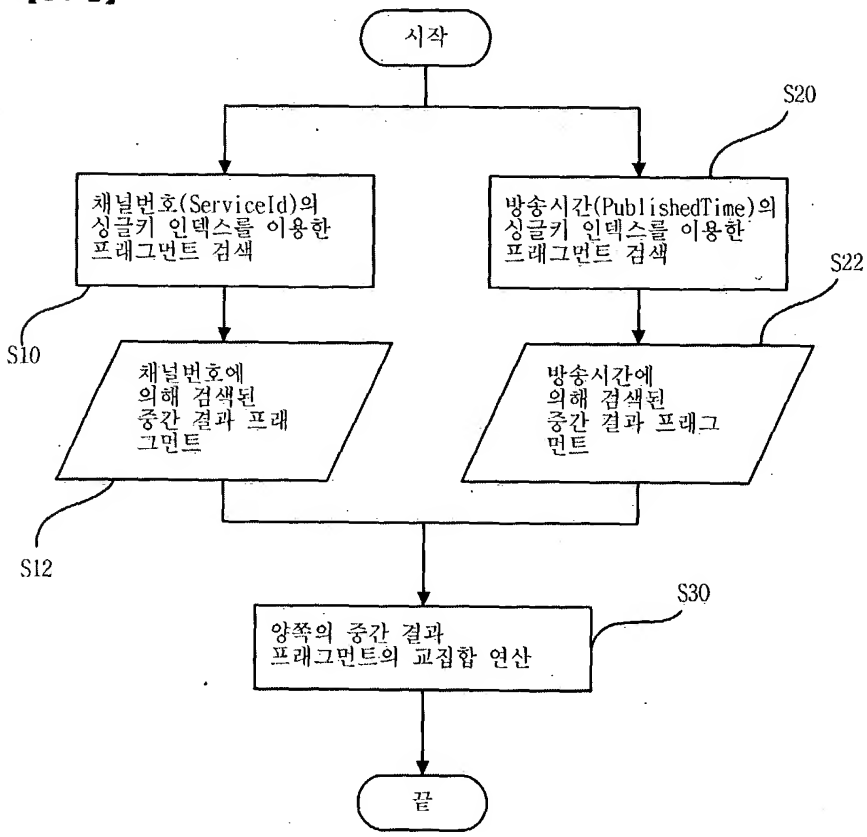
청구항 1에 있어서, 상기 키 인덱스 리스트 구조를 정의하는 단계는, 빈번히 참조되는 프래그먼트 타입에 대해 지정된 인코딩 값을 부여하는 단계를 포함하는 것을 특징으로 하는 디지털 컨텐츠 메타데이터 프래그먼트에 대한 복합 조건 검색 방법.

【도면】

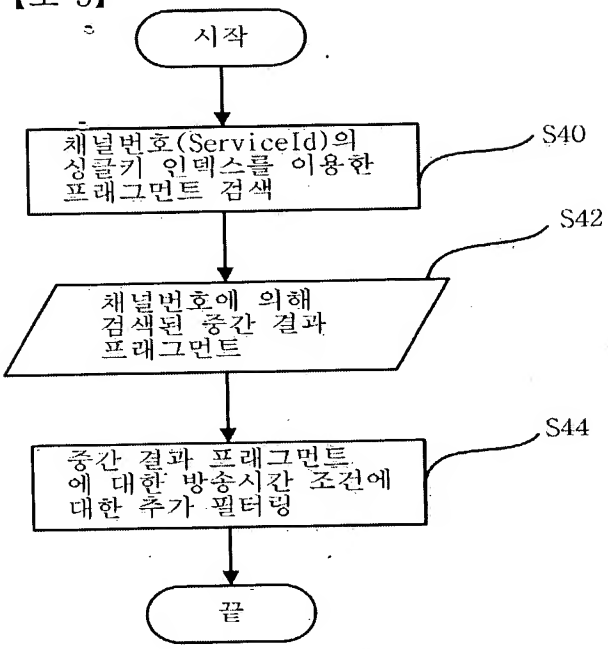
【도 1】


Today
9:00p
9:30p
10:00p
507 HBOF
508 HBFW
512 MAX
513 MMAX
514 MaxW
HD
Hocus Pocus
Vertical Limit - Movie
HD
American Be...

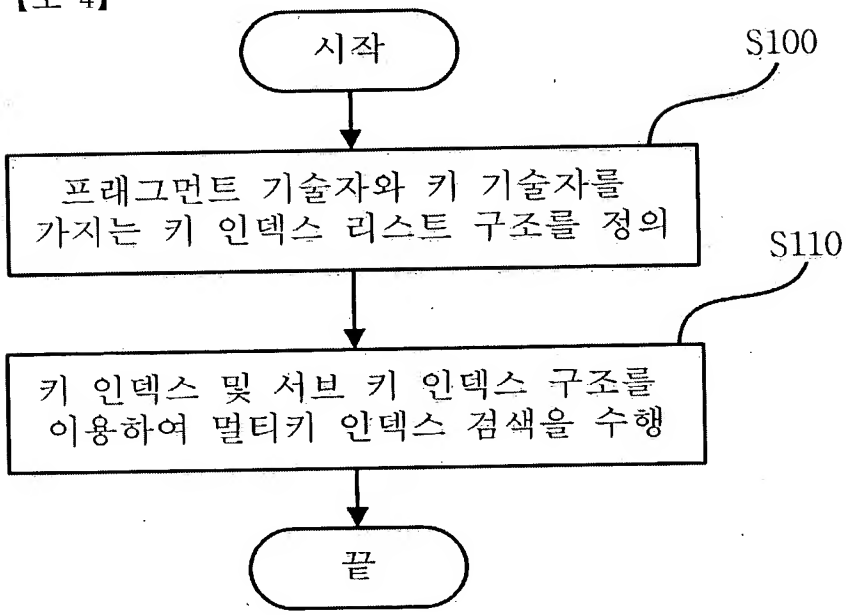
【도 2】


시작
S20
채널번호(ServiceId)의 싱글키 인덱스를 이용한 프래그먼트 검색
방송시간(PublishedTime)의 싱글키 인덱스를 이용한 프래그먼트 검색
S22
S10
채널번호에 의해 검색된 중간 결과 프래그먼트
방송시간에 의해 검색된 중간 결과 프래그먼트
S12
S30
양쪽의 중간 결과 프래그먼트의 교집합 연산
끝

【도 3】


시작
채널번호(ServiceId)의 싱글키 인덱스를 이용한 프래그먼트 검색
S40
채널번호에 의해 검색된 중간 결과 프래그먼트
S42
중간 결과 프래그먼트에 대한 방송시간 조건에 대한 추가 필터링
S44
끝

【도 4】


시작
S100
프래그먼트 기술자와 키 기술자를 가지는 키 인덱스 리스트 구조를 정의
S110
키 인덱스 및 서브 키 인덱스 구조를 이용하여 멀티키 인덱스 검색을 수행
끝

(Translation)

KOREAN INTELLECTUAL PROPERTY OFFICE

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

Application Number: 10-2002-0062913

Date of Application: October 15, 2002

Applicant(s): Samsung Electronics Co., Ltd.

Dated this 7th day of July, 2003

Commissioner (Seal)